UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica

Sicurezza dei Dati

ML Test Suite

A Simple webapp for testing Machine Learning models

Candidati:

Antonio Garofalo

Fulvio Serao

Professori:

Prof. Alfredo De Stantis

Prof. Christian Esposito

ANNO ACCADEMICO 2023/2024

Indice

Capitolo 1

Introduzione

1.1 Panoramica del progetto

Il progetto **ML Test Suite** è una suite di test avanzata progettata per valutare e migliorare la robustezza dei modelli di machine learning contro attacchi avversariali. Utilizzando la potente libreria Adversarial Robustness Toolkit (ART), la suite fornisce un set completo di strumenti e metriche per analizzare e mitigare le vulnerabilità dei modelli.

La libreria ART consente di generare esempi avversariali e di testare la capacità del modello di resistere a tali input. La suite è compatibile con vari framework di machine learning, tra cui TensorFlow, PyTorch e scikit-learn, e supporta diverse categorie di attacchi e tecniche difensive. La suite è progettata per essere integrata facilmente nei flussi di lavoro esistenti e offre una serie di metriche dettagliate per misurare l'efficacia delle strategie difensive e la robustezza complessiva dei modelli.

1.2 Obiettivi del progetto

L'obiettivo principale di **ML Test Suite** è fornire uno strumento completo per testare e migliorare la sicurezza dei modelli di machine learning. Questo progetto si propone di:

- **Effettuare il login** con MetaMask per accedere all'applicativo e gestire in sicurezza l'accesso dell'utente (futura implementazione con blockchain).
- **Testare il modello** dell'utente contro una varietà di attacchi avversariali caricando il proprio modello e completando la configurazione necessaria per il testing.
- **Applicare le difese** applicando tecniche difensive specifiche e analizzando come queste influenzano la resilienza del modello agli attacchi.
- **Analizzare i risultati** dell'evaluation dei modelli per comprendere la robustezza e le vulnerabilità, utilizzando metriche dettagliate fornite dal sistema e sfruttare tali conoscenze per applicare strategie di miglioramento sul proprio modello.

Il progetto si concentra su una valutazione approfondita e su una robustezza maggiore, offrendo agli sviluppatori e ai ricercatori gli strumenti necessari per garantire che i modelli di machine learning siano più sicuri e affidabili.

1.3 Attacco vs. Difesa

Ecco una rappresentazione del sistema attacco vs. difesa (credits to ART):

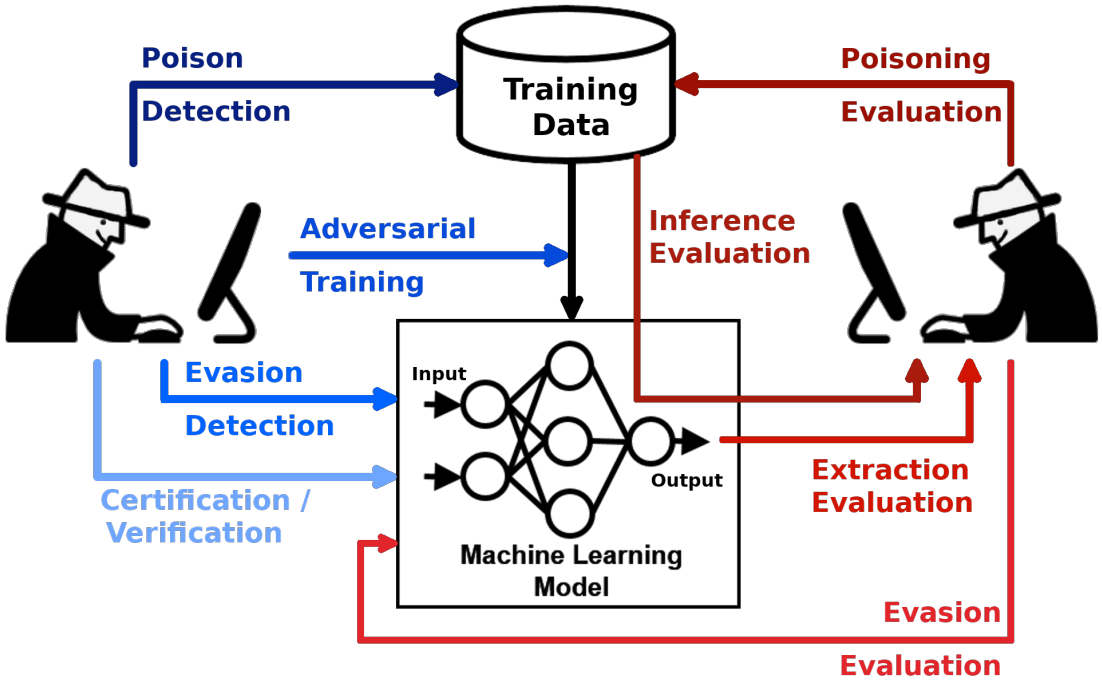

Figura 1.1: Esempio dimostrativo di attacco e difesa

Capitolo 2

Adversarial Robustness Toolkit (ART)

2.1 Intdorduzione

L'Adversarial Robustness Toolkit (ART)[1] è una libreria open-source sviluppata da IBM che fornisce strumenti avanzati per la creazione di modelli di machine learning robusti agli attacchi avversari. ART supporta una vasta gamma di attacchi e difese, permettendo ai ricercatori e agli sviluppatori di valutare la robustezza dei loro modelli e implementare contromisure efficaci.

2.2 Integrazione degli Attacchi

2.2.1 Evasion

L'attacco di evasion mira a ingannare un modello inducendolo a fare previsioni errate con input modificati in modo impercettibile. Questi attacchi sono di particolare interesse nella sicurezza dei modelli di apprendimento automatico, in quanto possono compromettere l'integrità e l'affidabilità del sistema. Tra le tecniche di attacco di evasion più studiate e utilizzate troviamo l'*FGM (Fast Gradient Method)* e il *PGD (Projected Gradient Descent)*.

Fast Gradient Method (FGM)

Il Fast Gradient Method (FGM) è uno dei metodi di attacco più semplici e veloci per generare esempi avversari. È stato proposto da Ian Goodfellow et al. nel 2014. Il principio di base dell'FGM è quello di sfruttare i gradienti del modello di rete neurale per trovare la direzione che massimizza l'errore di classificazione.

- **Definizione**: L'attacco FGM modifica leggermente l'input originale x aggiungendo una piccola perturbazione η nella direzione del gradiente della funzione di perdita rispetto all'input.
- **Formula**:

  $$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

  Dove:

  - $\tilde{x}$ è l'input malevolo generato.
  - ϵ è un piccolo parametro che controlla l'entità della perturbazione.
  - $\nabla_x J(\theta, x, y)$ è il gradiente della funzione di perdita J rispetto all'input x.
  - $\text{sign}(\cdot)$ è il segno del gradiente.
- **Vantaggi**: FGM è veloce da computare e richiede solo un passaggio di backpropagation.

- **Svantaggi**: Essendo un metodo di attacco di primo ordine, potrebbe non essere efficace contro modelli robusti o contro difese avanzate.

Projected Gradient Descent (PGD)

Il Projected Gradient Descent (PGD) è una generalizzazione iterativa dell'FGM, considerata una delle tecniche di attacco più potenti e versatili. È stato introdotto da Madry et al. nel 2017 come parte di uno studio approfondito sulla robustezza avversaria.

- **Definizione**: PGD applica iterativamente perturbazioni all'input originale, proiettando l'input perturbato nel vicino ϵ-ball attorno all'input originale per assicurarsi che la perturbazione rimanga impercettibile.
- **Formula**:

$$\tilde{x}^{(t+1)} = \Pi_{x+\mathcal{S}} \left(\tilde{x}^{(t)} + \alpha \cdot \text{sign}(\nabla_x J(\theta, \tilde{x}^{(t)}, y)) \right)$$

  Dove:

  - $\tilde{x}^{(t)}$ è l'input malevolo alla t-esima iterazione.
  - α è il passo dell'aggiornamento della perturbazione.
  - $\Pi_{x+\mathcal{S}}(\cdot)$ rappresenta la proiezione sull'insieme $\mathcal{S}$, che tipicamente è una palla L_∞ di raggio ϵ centrata sull'input originale x.

- **Procedura**:

  - Inizializzazione: $\tilde{x}^{(0)} = x$.
  - Per ogni iterazione t:
    - Calcola il gradiente della perdita rispetto all'input perturbato.
    - Aggiorna l'input perturbato aggiungendo una piccola perturbazione nella direzione del gradiente.
    - Proietta l'input perturbato nell'insieme consentito per mantenere la perturbazione impercettibile.

- **Vantaggi**: PGD è considerato un attacco di "forte malevolo" e può rompere difese che resistono a metodi di attacco più semplici come FGM.
- **Svantaggi**: È computazionalmente più costoso rispetto a FGM, poiché richiede molteplici iterazioni di aggiornamento.

2.2.2 Extraction

L'attacco di extraction mira a ricostruire o copiare un modello addestrato estraendo conoscenza dal modello bersaglio. Questo tipo di attacco può compromettere la proprietà intellettuale e la riservatezza del modello, consentendo a un malevolo di ottenere un modello funzionante simile senza accesso diretto ai dati di addestramento o ai parametri originali.

CopycatCNN

Il metodo CopycatCNN è un esempio di attacco di extraction in cui un malevolo addestra un modello sostitutivo, chiamato modello copycat, utilizzando il modello bersaglio come "oracolo" per etichettare un insieme di dati sintetici o raccolti autonomamente. Il modello copycat imita il comportamento del modello bersaglio, cercando di replicarne le prestazioni.

- **Definizione**: CopycatCNN utilizza il modello bersaglio per etichettare un insieme di input, che vengono poi utilizzati per addestrare il modello copycat.
- **Procedura**:
    1. **Raccolta dati**: L'malevolo raccoglie un insieme di input, che può essere sintetico o provenire da una distribuzione simile a quella del modello bersaglio.
    2. **Etichettatura**: Gli input raccolti vengono passati attraverso il modello bersaglio, che fornisce le etichette predette.
    3. **Addestramento**: Il modello copycat viene addestrato utilizzando gli input raccolti e le etichette fornite dal modello bersaglio.
    4. **Valutazione**: Il modello copycat viene valutato per verificare quanto bene riesce a replicare le prestazioni del modello bersaglio.
- **Vantaggi**: CopycatCNN permette di creare un modello sostitutivo con prestazioni simili a quelle del modello bersaglio senza conoscere i dettagli

interni del modello bersaglio o avere accesso ai dati originali di addestramento.

- **Svantaggi**: La qualità del modello copycat dipende dalla qualità e dalla quantità dei dati raccolti e dall'accuratezza delle predizioni del modello bersaglio.

2.2.3 Model Inversion

L'attacco di inversione del modello cerca di ricostruire i dati di input originali dai loro output, compromettendo la privacy dei dati. Questo tipo di attacco può rivelare informazioni sensibili sugli individui o sui dati utilizzati per addestrare il modello.

MIFace

Il metodo MIFace è un esempio di attacco di inversione del modello specificamente applicato al riconoscimento facciale. Utilizzando l'accesso ai punteggi o alle probabilità di output del modello di riconoscimento facciale, MIFace può ricostruire immagini facciali che sono simili agli input originali.

- **Definizione**: MIFace utilizza le risposte del modello di riconoscimento facciale per ricostruire un'immagine facciale che produce output simili.
- **Procedura**:
    1. **Inizializzazione**: Si parte con un'immagine iniziale casuale o una media di facce.
    2. **Ottimizzazione**: L'immagine viene iterativamente modificata per ridurre la differenza tra gli output del modello per l'immagine modificata e gli output per l'immagine target.
    3. **Aggiornamento**: L'aggiornamento dell'immagine viene eseguito utilizzando tecniche di ottimizzazione, come la discesa del gradiente, per minimizzare la funzione di perdita che misura la somiglianza degli output.
- **Vantaggi**: MIFace può rivelare informazioni dettagliate e potenzialmente sensibili sugli individui, compromettendo seriamente la privacy.
- **Svantaggi**: L'efficacia del metodo può variare a seconda della complessità del modello e della quantità di informazioni disponibili sugli output del modello.

2.2.4 Poisoning

L'attacco di poisoning consiste nell'inserire dati manipolati nel set di addestramento per compromettere il modello durante la fase di training. Questo tipo di attacco può causare malfunzionamenti del modello in fase di inferenza o inserire vulnerabilità specifiche che possono essere sfruttate successivamente.

Clean Label Backdoor

Il metodo Clean Label Backdoor è un attacco di poisoning in cui l'attaccante manipola i dati di addestramento in modo da non alterare le etichette, rendendo difficile rilevare la manipolazione. L'obiettivo è inserire una backdoor nel modello che possa essere attivata da specifici trigger durante l'inferenza.

- **Definizione**: Un attacco Clean Label Backdoor modifica gli input di addestramento in modo sottile, mantenendo le etichette corrette, così che le modifiche siano impercettibili durante l'addestramento.
- **Procedura**:
    1. **Selezione dei dati**: L'attaccante seleziona un sottoinsieme di dati di addestramento.
    2. **Inserimento del trigger**: L'attaccante inserisce un piccolo trigger nei dati selezionati (ad esempio, un piccolo pattern o perturbazione).
    3. **Addestramento**: Il modello viene addestrato su questi dati manipolati insieme ai dati originali.
    4. **Attivazione**: Durante l'inferenza, il modello riconoscerà il trigger e produrrà un output specifico voluto dall'attaccante.
- **Vantaggi**: Difficile da rilevare perché le etichette rimangono inalterate e i trigger sono spesso impercettibili.
- **Svantaggi**: La creazione di un trigger efficace può richiedere una conoscenza approfondita del modello e dei dati di addestramento.

Simple Backdoor

Il metodo Simple Backdoor è un attacco di poisoning in cui l'attaccante modifica gli input di addestramento e le loro etichette per inserire una backdoor nel modello. A differenza del Clean Label Backdoor, qui sia gli input che le etichette sono alterati.

- **Definizione**: Un attacco Simple Backdoor manipola gli input e le etichette di addestramento per creare una vulnerabilità specifica che può essere sfruttata tramite un trigger noto.
- **Procedura**:
  1. **Selezione dei dati**: L'attaccante seleziona un sottoinsieme di dati di addestramento.
  2. **Inserimento del trigger**: L'attaccante inserisce un trigger distintivo nei dati selezionati.
  3. **Modifica delle etichette**: Le etichette dei dati con il trigger vengono modificate per riflettere l'output desiderato dall'attaccante.
  4. **Addestramento**: Il modello viene addestrato sui dati manipolati insieme ai dati originali.
  5. **Attivazione**: Durante l'inferenza, il modello riconoscerà il trigger e produrrà l'output manipolato.
- **Vantaggi**: Più semplice da implementare rispetto al Clean Label Backdoor perché non richiede che le etichette rimangano inalterate.
- **Svantaggi**: Può essere più facilmente rilevato durante la verifica dei dati di addestramento a causa delle etichette modificate.

2.3 Integrazione delle Difese

2.3.1 Detector

I detector sono progettati per identificare gli input avversari. Questi strumenti analizzano gli input ricevuti dal modello per determinare se sono stati manipolati in modo malevolo. Una delle tecniche di difesa più conosciute in questo ambito è la *Activation Defense*.

Activation Defense

L'Activation Defense sfrutta le attivazioni interne del modello per rilevare input avversari. L'idea di base è che gli input avversari generano attivazioni interne diverse rispetto agli input legittimi.

- **Definizione**: L'Activation Defense monitora le attivazioni nei livelli interni del modello per identificare anomalie che possono indicare la presenza di input avversari.
- **Procedura**:
    1. **Raccolta delle attivazioni**: Durante l'addestramento, vengono raccolte e memorizzate le attivazioni dei livelli interni per gli input legittimi.
    2. **Creazione del profilo**: Viene creato un profilo delle attivazioni legittime, che serve come riferimento per il rilevamento delle anomalie.
    3. **Rilevamento delle anomalie**: Durante l'inferenza, le attivazioni degli input ricevuti vengono confrontate con il profilo delle attivazioni legittime. Gli input che deviano significativamente dal profilo vengono considerati potenzialmente avversari.
- **Vantaggi**: Può rilevare una vasta gamma di attacchi avversari, poiché si basa sulle caratteristiche interne del modello piuttosto che sugli input stessi.

- **Svantaggi**: Richiede una quantità significativa di risorse per la raccolta e l'analisi delle attivazioni, e la sua efficacia dipende dalla qualità del profilo delle attivazioni legittime.

2.3.2 Trainer

I trainer migliorano la robustezza del modello durante la fase di addestramento. Questi metodi sono progettati per rendere i modelli di machine learning più resistenti agli attacchi avversari, migliorando la loro capacità di gestire input manipolati malevolmente.

Adversarial Trainer

L'Adversarial Trainer è una tecnica che incorpora esempi avversari nel processo di addestramento del modello. L'obiettivo è rendere il modello meno sensibile agli attacchi avversari aumentando la sua capacità di generalizzare bene anche su input perturbati.

- **Definizione**: L'Adversarial Trainer genera esempi avversari durante l'addestramento e li utilizza insieme agli esempi legittimi per addestrare il modello.
- **Procedura**:
    1. **Generazione degli esempi avversari**: Durante ogni iterazione di addestramento, vengono generati esempi avversari utilizzando metodi come FGM o PGD.
    2. **Addestramento con esempi avversari**: Gli esempi avversari vengono inclusi nel set di addestramento insieme agli esempi legittimi.
    3. **Aggiornamento del modello**: Il modello viene aggiornato per minimizzare la funzione di perdita sia sugli esempi legittimi che su quelli avversari.
- **Vantaggi**: Migliora significativamente la robustezza del modello contro gli attacchi avversari, rendendolo più resistente alle perturbazioni.
- **Svantaggi**: L'addestramento con esempi avversari richiede maggiori risorse computazionali e può rallentare il processo di addestramento.

2.3.3 Transformer

I transformer modificano i dati per ridurre l'efficacia degli attacchi. Questi metodi sono progettati per trasformare gli input in modo da diminuire la probabilità che un attacco malevolo abbia successo, mantenendo al contempo l'integrità e l'utilità dei dati.

STRIP (STRong Intentional Perturbation)

Il metodo STRIP (*STRong Intentional Perturbation*) è una tecnica utilizzata per rilevare e mitigare gli attacchi avversari. STRIP identifica gli input avversari introducendo perturbazioni intenzionali e osservando la variazione nelle predizioni del modello.

- **Definizione**: STRIP introduce perturbazioni intenzionali agli input e verifica la consistenza delle predizioni del modello. Se le predizioni variano significativamente, l'input è probabilmente malevolo.
- **Procedura**:
  1. **Applicazione delle perturbazioni**: Ogni input viene modificato più volte introducendo perturbazioni intenzionali (ad esempio, cambiamenti di pixel, rumore aggiuntivo).
  2. **Inferenza**: Il modello effettua predizioni su ciascuna versione perturbata dell'input.
  3. **Analisi delle predizioni**: Le predizioni risultanti vengono analizzate per verificarne la consistenza. Input legittimi tendono a produrre predizioni stabili nonostante le perturbazioni, mentre input avversari mostrano predizioni incoerenti.
- **Vantaggi**: STRIP è efficace nel rilevare una vasta gamma di attacchi avversari senza richiedere modifiche al modello di base.
- **Svantaggi**: L'applicazione di perturbazioni multiple e l'analisi delle predizioni richiedono risorse computazionali aggiuntive, che possono rallentare il processo di inferenza.

2.3.4 Post Processor e Pre Processor

I post processor e i pre processor sono utilizzati rispettivamente prima e dopo la predizione del modello per incrementare la robustezza. Questi metodi aiutano a migliorare la resistenza del modello agli attacchi avversari attraverso trasformazioni specifiche degli input o degli output.

Post Processor: Reverse Sigmoid

Il *Reverse Sigmoid* è una tecnica di post processing utilizzata per mitigare l'effetto degli attacchi avversari applicando una trasformazione inversa alla funzione sigmoide sugli output del modello.

- **Definizione**: Il Reverse Sigmoid trasforma gli output del modello utilizzando una funzione inversa della sigmoide per ridurre l'impatto delle perturbazioni avversarie sugli output finali.
- **Procedura**:
  1. **Calcolo degli output**: Gli output originali del modello vengono calcolati normalmente.
  2. **Applicazione della trasformazione**: Viene applicata la trasformazione inversa della sigmoide agli output del modello.
  3. **Predizione finale**: Gli output trasformati vengono utilizzati per determinare la predizione finale del modello.
- **Vantaggi**: Aiuta a ridurre l'impatto delle perturbazioni avversarie mantenendo la robustezza del modello.
- **Svantaggi**: Può introdurre complessità computazionale aggiuntiva nel processo di inferenza.

Pre Processor: Total Variation Minimization

La *Total Variation Minimization* è una tecnica di pre processing utilizzata per ridurre il rumore negli input, rendendo più difficile per gli attacchi avversari influenzare il modello.

- **Definizione**: La Total Variation Minimization riduce la variazione totale nell'immagine di input, attenuando le perturbazioni avversarie che potrebbero compromettere la predizione del modello.
- **Procedura**:
  1. **Calcolo della variazione totale**: Viene calcolata la variazione totale dell'immagine di input.
  2. **Riduzione della variazione**: Si applica un algoritmo di ottimizzazione per minimizzare la variazione totale, rimuovendo le perturbazioni avversarie.
  3. **Input trasformato**: L'immagine trasformata, con variazione ridotta, viene utilizzata come input per il modello.
- **Vantaggi**: Effettivo nel rimuovere il rumore e le perturbazioni avversarie, migliorando la robustezza del modello.
- **Svantaggi**: Può richiedere un tempo di calcolo significativo, specialmente per immagini di grandi dimensioni.

Capitolo 3

Tecnologie Adoperate

3.1 Frontend

3.1.1 React

React[2] è una libreria JavaScript[3] open-source sviluppata da Facebook, utilizzata per la creazione di interfacce utente (UI). È particolarmente apprezzata per la sua capacità di rendere più semplice il processo di sviluppo di componenti UI riutilizzabili. Le caratteristiche principali di React includono:

- **Component-based**: React utilizza un approccio basato su componenti, dove ogni parte dell'interfaccia utente è rappresentata come un componente isolato. Questo favorisce una gestione più modulare, facilitando la manutenzione e il riutilizzo del codice.
- **Virtual DOM**: React utilizza un Virtual DOM (Document Object Model) per migliorare le performance delle applicazioni. Il Virtual DOM è una rappresentazione leggera della struttura dell'interfaccia utente che React mantiene in memoria e confronta con il DOM effettivo. Questo approccio consente a React di aggiornare solo le parti del DOM che sono cambiate, rendendo l'applicazione più veloce ed efficiente.
- **JSX: React introduce JSX (JavaScript XML)**, una sintassi che permette di scrivere codice HTML all'interno di JavaScript. Questo permette

agli sviluppatori di definire facilmente la struttura dell'interfaccia utente direttamente nel codice JavaScript, migliorando la leggibilità e facilitando la manipolazione dinamica dei componenti.

3.1.2 Vite

Vite[4] è un framework di sviluppo front-end moderno e leggero, progettato per facilitare la creazione di applicazioni web veloci. Alcune caratteristiche salienti di Vite includono:

- **Dev server veloce**: Vite è noto per il suo dev server veloce che supporta il caricamento rapido dei moduli. Utilizza ES Modules nativi del browser durante lo sviluppo, evitando la necessità di compilazioni e bundling pesanti durante il processo di sviluppo.
- **Supporto per ES Modules**: Vite supporta nativamente ES Modules, consentendo agli sviluppatori di importare moduli JavaScript direttamente all'interno del browser. Questo approccio è più efficiente rispetto agli strumenti tradizionali che richiedono la creazione di bundle statici.
- **Plugin System**: Vite offre un sistema di plugin estensibile che permette agli sviluppatori di personalizzare e estendere facilmente il funzionamento del framework secondo le proprie esigenze. Questo include supporto per plugin che migliorano il supporto di vari tipi di file (come TypeScript, React, CSS, ecc.) e ottimizzazioni per le prestazioni.

3.1.3 Integrazione di React con Vite

React e Vite sono complementari e possono essere integrati insieme per sfruttare al meglio le loro rispettive caratteristiche. Per utilizzare React con Vite, è possibile configurare Vite per supportare React utilizzando il plugin ufficiale @vitejs/plugin-react. Questo plugin abilita il supporto per JSX e facilita l'integrazione di React all'interno di un progetto Vite senza la necessità di configurazioni complesse.

In sintesi, React è una libreria per la creazione di interfacce utente component-based, nota per il suo Virtual DOM e l'uso di JSX. Vite, d'altra parte, è un framework di sviluppo front-end moderno, veloce e leggero, ideale per il caricamento rapido dei moduli durante lo sviluppo. Insieme, React e Vite consentono di creare applicazioni web moderne e performanti, migliorando la produttività degli sviluppatori e l'esperienza dell'utente finale.

3.2 Backend

3.2.1 FastAPI

Il backend del nostro WTFunko Store è stato sviluppato utilizzando FastAPI[5], un moderno framework web per Python[6] che permette di creare API RESTful in modo rapido ed efficiente. FastAPI è stato scelto per le sue eccellenti performance, la facilità d'uso e la capacità di generare automaticamente documentazione interattiva.

Caratteristiche principali di FastAPI

- **Facilità di utilizzo**: FastAPI permette di definire le API utilizzando le annotazioni di tipo di Python. Questo rende il codice più leggibile e aiuta a prevenire errori, migliorando la produttività degli sviluppatori.
- **Performance elevate**: FastAPI è basato su Starlette per il web e Pydantic per la gestione dei dati. Questo consente al framework di offrire performance molto elevate, paragonabili a quelle di Node.js e Go, rendendolo adatto per applicazioni web ad alta intensità di traffico.
- **Validazione automatica dei dati**: Grazie a Pydantic, FastAPI esegue automaticamente la validazione dei dati di input e di output, assicurando che i dati rispettino le specifiche definite dagli sviluppatori.
- **Documentazione automatica**: FastAPI genera automaticamente documentazione interattiva delle API utilizzando OpenAPI e Swagger UI. Questo facilita la comprensione e l'uso delle API da parte degli sviluppatori e dei team di integrazione.
- **Supporto per l'asincronia**: FastAPI supporta nativamente le operazioni asincrone utilizzando async e await, permettendo di gestire un numero elevato di richieste concorrenti in modo efficiente.

Architettura dell'applicazione

Il progetto backend è strutturato in modo modulare per facilitare la manutenzione e l'espansione futura. Di seguito una panoramica delle principali componenti:

- **App principale**: L'app principale rappresenta il cuore dell'applicazione, dove viene creata l'istanza di FastAPI, il framework utilizzato per costruire l'API web. In questa fase, vengono configurati i parametri principali dell'applicazione e vengono registrate le varie route che definiscono gli endpoint dell'API. La struttura modulare di FastAPI consente di mantenere il codice organizzato e scalabile, facilitando la gestione delle richieste HTTP, l'elaborazione dei dati e la risposta agli utenti. La creazione e configurazione dell'istanza di FastAPI includono anche la definizione delle impostazioni globali e la configurazione dei componenti essenziali, come i middleware e le dipendenze comuni.

- **Router**: I router sono componenti fondamentali per l'organizzazione e la gestione degli endpoint dell'API. In FastAPI, i router permettono di separare le diverse funzionalità dell'applicazione in moduli distinti. Ad esempio, è possibile avere router dedicati alla gestione dei prodotti, degli ordini e all'autenticazione degli utenti. Questa separazione aiuta a mantenere il codice pulito e modulare, migliorando la manutenibilità e facilitando la navigazione tra le diverse parti del codice. Ogni router può definire endpoint specifici per le operazioni CRUD (Create, Read, Update, Delete) e può essere facilmente esteso o modificato senza influire sugli altri router.

- **Modelli**: I modelli in FastAPI sono definiti utilizzando Pydantic, una libreria per la validazione dei dati basata su Python. I modelli Pydantic permettono di definire chiaramente la struttura dei dati di input e output per gli endpoint dell'API. Questi modelli garantiscono che i dati siano validi e corretti prima di essere elaborati, riducendo il rischio di errori e migliorando la robustezza dell'applicazione. Ogni modello può includere regole di validazione, tipi di dati e valori predefiniti, assicurando che le richieste

siano coerenti con le aspettative dell'API. L'uso di Pydantic non solo migliora la qualità dei dati, ma semplifica anche la documentazione automatica dell'API, che FastAPI genera basandosi sui modelli definiti.

- **MetaMask**: Per l'autenticazione, l'applicazione utilizza MetaMask[7], un'estensione del browser che funge da portafoglio Ethereum. MetaMask offre un'interfaccia intuitiva per interagire con la blockchain Ethereum, consentendo agli utenti di gestire i loro portafogli e di firmare transazioni direttamente dal loro browser. Attualmente, MetaMask viene utilizzato principalmente per autenticare gli utenti all'interno dell'applicazione, sfruttando la sua integrazione con la blockchain come preparazione per futuri sviluppi di smart contracts e transazioni in blockchain.

Vantaggi dell'uso di FastAPI

- **Velocità di sviluppo**: La sintassi semplice e le funzionalità integrate di FastAPI accelerano il processo di sviluppo, riducendo il tempo necessario per implementare nuove funzionalità.
- **Performance**: La gestione asincrona delle richieste e l'efficienza del framework garantiscono alte performance, rendendo FastAPI ideale per applicazioni scalabili e ad alta intensità di traffico.
- **Documentazione**: La generazione automatica di documentazione facilita la collaborazione tra i team e migliora la comunicazione con gli sviluppatori che integrano le API.

3.2.2 Sicurezza

La sicurezza delle API è essenziale per proteggere i dati sensibili degli utenti e garantire che solo gli utenti autorizzati possano accedere a determinate risorse. Nel nostro backend FastAPI, abbiamo implementato diverse misure di sicurezza, tra cui il middleware CORS, la suddivisione delle API in base ai livelli di accesso e l'uso di token per l'autenticazione.

Middleware CORS (Cross-Origin Resource Sharing)

Il middleware CORS è utilizzato per controllare quali domini possono accedere alle API del server. Questo è particolarmente importante per le applicazioni web che fanno richieste AJAX a un dominio diverso da quello da cui è stata caricata l'applicazione.

Accesso alle API tramite Random Access Token

In una futura versione dell'applicativo è prevista l'integrazione di accessi alle API del server tramite la condivisione con un access token generato randomicamente.

3.2.3 Documentazione API

Nella creazione delle route per le nostre REST API abbiamo pensato di avere un accesso. Inoltre è stato implementata una route di default del server "**/docs**" che permette di avere una visione d'insieme di tutti gli endpoint REST API e la possibilità di utilizzarli (secondo un criterio di accesso).

Di seguito sono mostrati tutti gli endpoint:

Endpoint	**Metodo**	**Sommario**	**Tipo**
/attack/evasion/{`attack_type`}	POST	Evasion attack perform route.	Attack
/attack/extraction/{`attack_type`}	POST	Extraction attack perform route.	Attack
/attack/inference/{`attack_type`}	POST	Inference attack perform route.	Attack
/attack/poisoning/{`attack_type`}	POST	Poisoning attack perform route.	Attack
/defense/detector/{`defense_type`}	POST	Detector defense perform route.	Defense
/defense/postprocessor/{`defense_type`}	POST	Postprocessor defense perform route.	Defense
/defense/preprocessor/{`defense_type`}	POST	Preprocessor defense perform route.	Defense
/defense/trainer/{`defense_type`}	POST	Trainer defense perform route.	Defense
/defense/transformer/{`defense_type`}	POST	Transformer defense perform route.	Defense
/upload/model	POST	Upload a model	Upload
/upload/dataset	POST	Upload a dataset	Upload

Tabella 3.1: API del Server FastAPI per attacchi, difese, e upload

Capitolo 4

Implementazione

In questa sezione, mostreremo il lavoro svolto per il progetto ML Test Suite. In particolare mostreremo le schermate principali e una schermata DEMO, nella repository su GitHub potete trovare anche un powerpoint illustrativoPer ogni screenshot della webapp, forniremo una breve descrizione e spiegazione.

4.1 Use Case

4.1.1 Attori

1. **Utente** - L'attore principale che testa si interfaccia con il sistema.
2. **Metamask** - Componente/Attore secondario che si occupa di interfacciarsi con l'SDK per l'autenticazione.
3. **ML Test Suite** - Componente/Attore secondario che si occupa di gestire gli input/output dell'utente e la logica di sistema.

4.1.2 Descrizione

L'utente interagisce con la ML Test Suite per eseguire test di robustezza su modelli di apprendimento automatico. L'uso della suite comprende il caricamento di modelli, la scelta di dataset, la configurazione di parametri e l'esecuzione di vari test (attacchi, difese, modelli).

4.1.3 Precondizioni

- L'utente deve essere autenticato nel sistema. - L'utente deve avere accesso ai modelli e ai dataset necessari per il test.

4.1.4 Postcondizioni

- L'utente ottiene i risultati dei test di robustezza eseguiti sui modelli.

4.1.5 Passi Principali

1. **Login**: L'utente accede al sistema.
2. **Upload Model**: L'utente carica il modello nel sistema, che può essere in formato compilato o non compilato (.h5).
3. **Upload Dataset**: L'utente carica il dataset da utilizzare per i test.
4. **Dataset Choice**: L'utente sceglie quale dataset utilizzare.
5. **Params Choice**: L'utente configura i parametri per i test.
6. **Test Attack**: L'utente esegue test di attacco per valutare la robustezza del modello contro attacchi avversari.
7. **Test Defense**: L'utente esegue test di difesa per verificare l'efficacia delle difese implementate.
8. **Test Model**: L'utente esegue test generali sul modello.

4.1.6 Eccezioni

1. **Errore di Caricamento**: Se il modello o il dataset non vengono caricati correttamente, viene visualizzato un messaggio di errore.
2. **Errore di Autenticazione**: Se l'utente non riesce ad autenticarsi, non può accedere alle funzionalità della suite.

3. **Errori generici di sistema**: Se il sistema si presenta in uno stato anomalo comunica all'utente (tramite opportuna console) lo stato e procede con lo shutdown.

4.1.7 Use Case Diagram

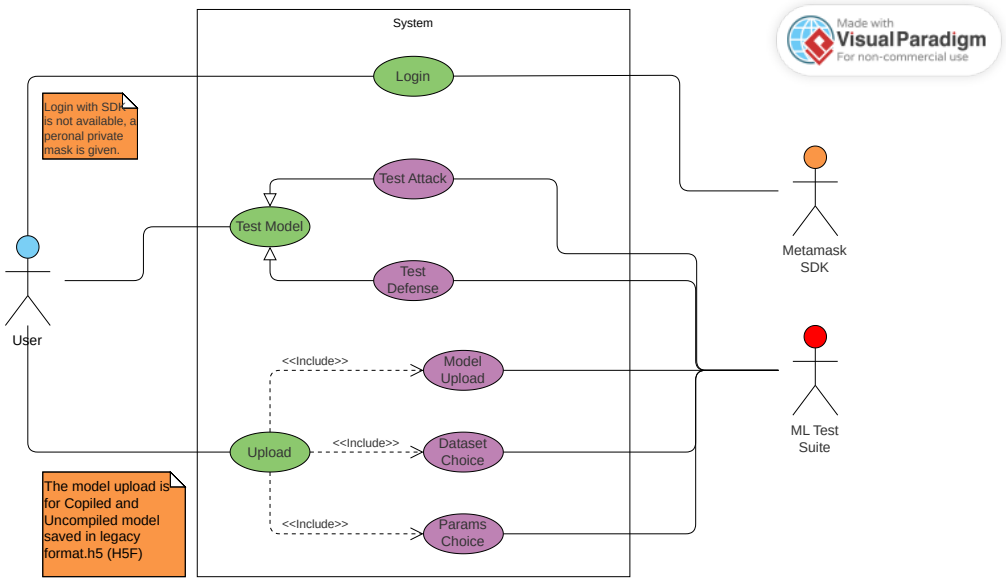

Figura 4.1: Use Case

4.2 Illustrazioni

4.2.1 Logging e Terminale

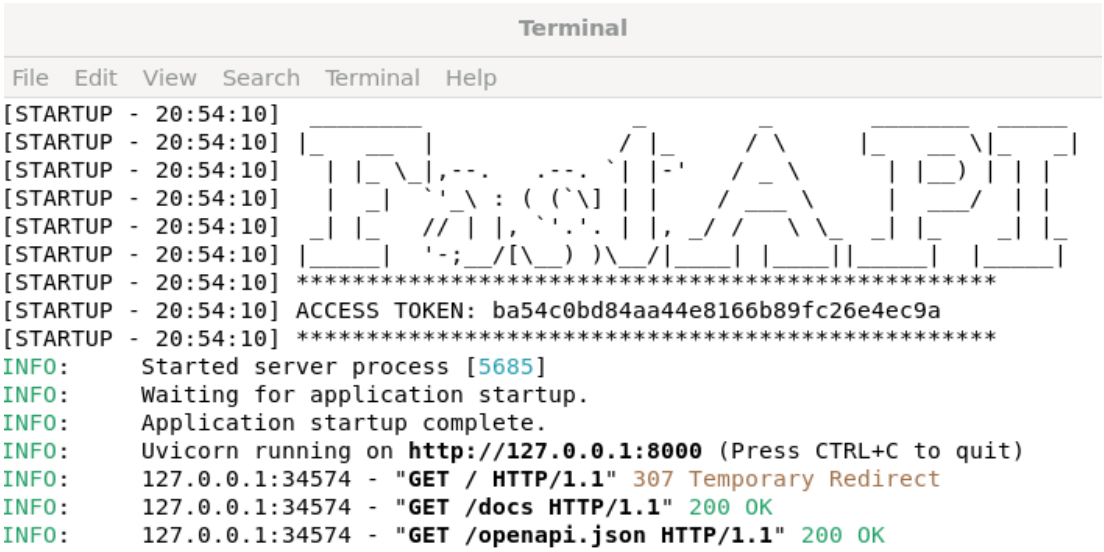

Figura 4.2: Terminale

Ecco lo screen di startup della console del server, da qui verranno (per poi essere salvate su un file apposito) scritte tutte le operazioni del server mediante l'utilizzo di una classe personalizzata ad-hoc per il logging.

```python
class Logger(object):
_instance = None
_log = None

def __new__(cls, filename=None, directory=None):
    if cls._instance is None:
        if filename is None or directory is None:
            raise ValueError("Filename and directory must be provided for the first
                initialization.")
        cls._instance = super(Logger, cls).__new__(cls)
        cls._instance._initialize(filename, directory)
    return cls._instance

def _initialize(self, filename=None, directory=None):
    if filename is not None and directory is not None:
        if not isinstance(filename, str) or not isinstance(directory, str):
            self.write("Logging", "Error: File name and directory must be strings.")
            self._log = None
        elif any(c in r'\/:*?"<>|' for c in filename):
            self.write("Logging", "Error: File name contains invalid characters.")
            self._log = None
        else:
            if not os.path.exists(f"{directory}"):
                os.makedirs(f"{directory}")
            curr_date = dt.now().isoformat().replace(':', '-')
            self._log = open(f"{directory}/{filename}_{curr_date}.log", "x")
    else:
        self._log = None

def write(self, tag, text):
    curr_time = dt.now().strftime("%H:%M:%S")
    log_message = f"[{tag} - {curr_time}] {text}\n"
    sys.stdout.write(log_message)

    if self._log is not None:
        self._log.write(log_message)
        self.flush()

def flush(self):
    if self._log is not None:
        self._log.flush()
```

Listing 4.1: Logger Class

4.2.2 FastAPI Docs Console

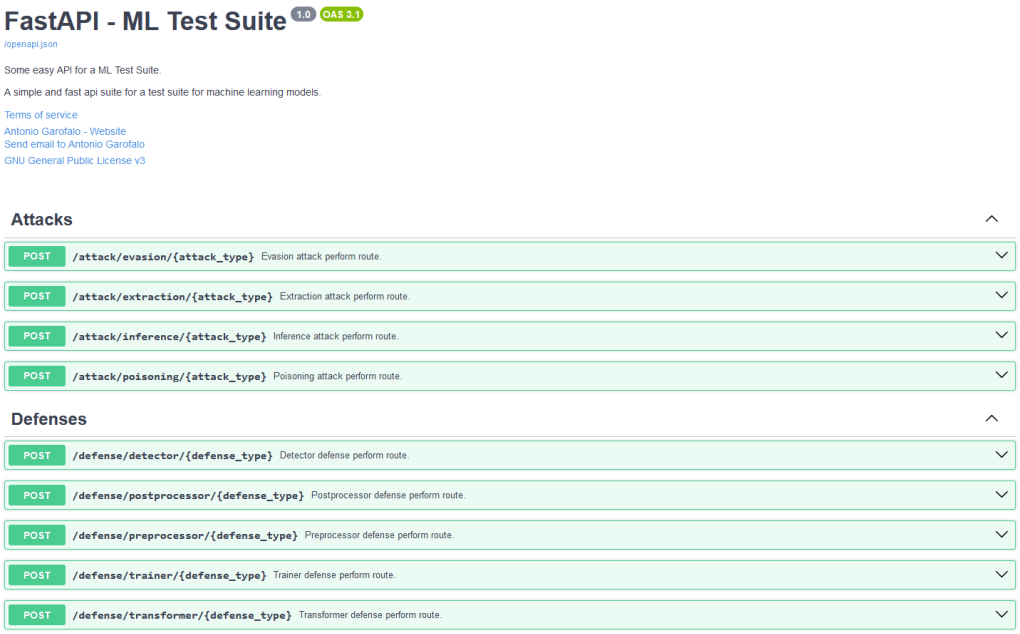

Figura 4.3: FasAPI docs page

La route `/docs` di FastAPI fornisce una documentazione interattiva dell'API usando Swagger UI. Permette agli sviluppatori di esplorare e testare le endpoint dell'API direttamente dal browser. La documentazione è generata automaticamente e si aggiorna con le modifiche all'API, facilitando la comprensione e il debugging.

4.2.3 An example of REST API tester suite

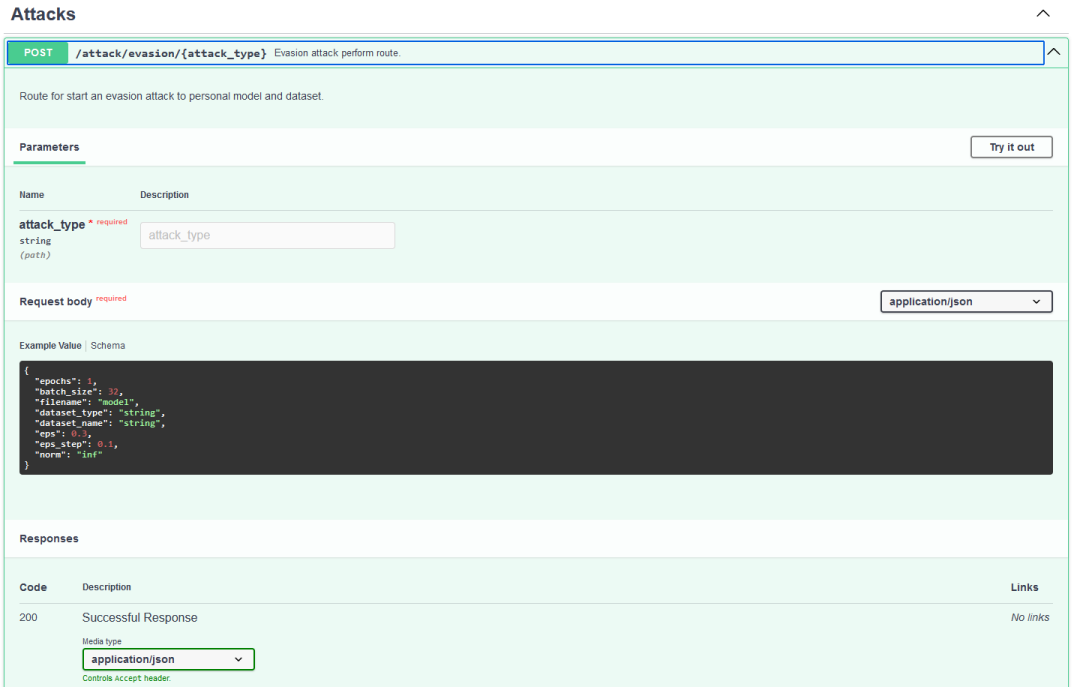

Figura 4.4: FGM REST API tester

Nella suite Swagger UI di FastAPI da noi personalizzata è possibile visualizzare un anteprima della request che il server si aspetta, la response che può ritornare, eventuali schemi di errori o modelli alternativi e i parametri che accetta, compresi di tipo e validator.

4.2.4 Homepage

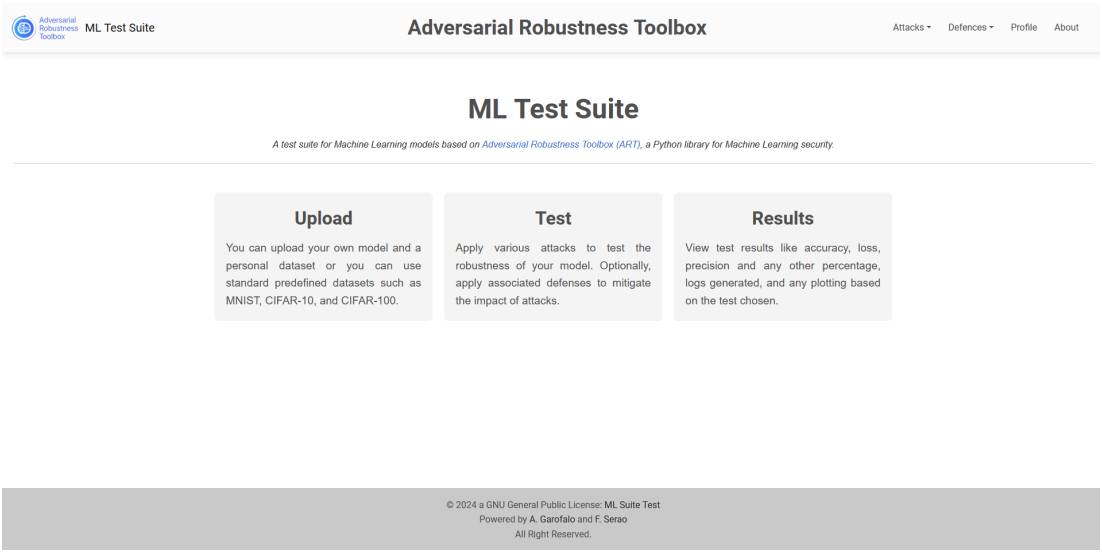

Figura 4.5: Homepage

Nella homepage è possibile accedere al profilo collegato con Metamask, alla repository GitHub del progetto cliccando su "About" e alle due liste di attacchi e difesa disponibili per apssare in modo semplice e intuitivo alle pagine di test. La homepage descrive brevemente il progetto.

4.2.5 An example of Testing page

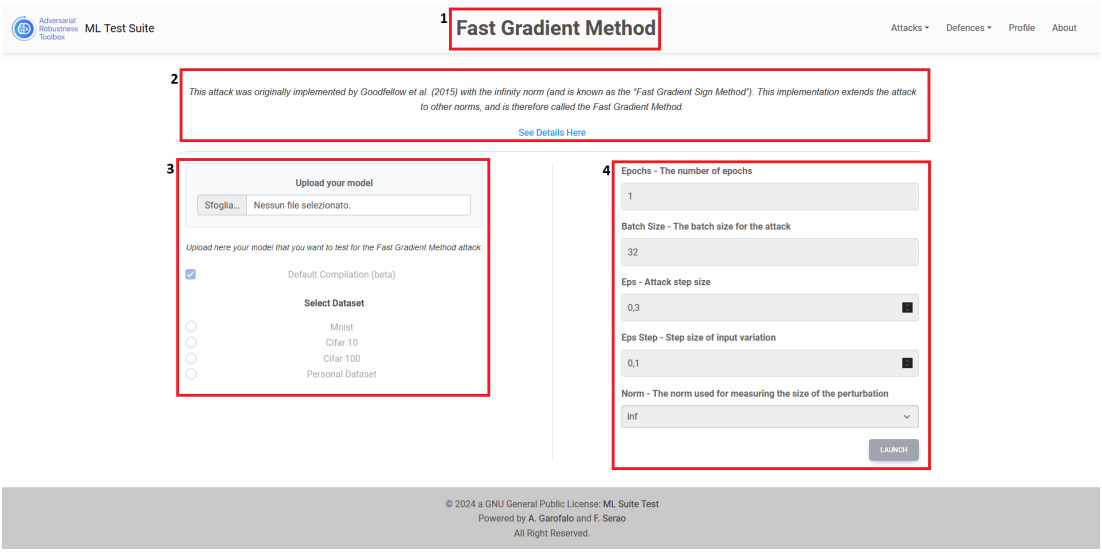

Figura 4.6: Pagina di test d'esempio

In questa schermata di test (Fast Gradient Method (FGM)) viene mostrato un esempio di test page. L'applicativo è semplice ed elegante e di facile intuizione puntato tutto sul controllo di input, descrizione dettagliata e limitazione delle azioni arbitrarie dell'utente. Possiamo notare:

1. Il nome dell'attacco/difesa.
2. Breve descrizione dell'attacco/difesa con link diretto al paper ufficiale (pubblicazione IBM).
3. Sezione di upload per permettere all'utente di caricare il proprio modello, scegliere un dataset e in futuro creare o compilare il proprio modello con parametri built-in.
4. Sezione di parametrizzazione del test con controlli di validazione e popup di conferma per operazioni sensibili.

Capitolo 5

Conclusioni

Il progetto di sviluppo della **ML Test Suite** utilizzando l'Adversarial Robustness Toolbox (ART) ha permesso di ottenere un sistema robusto e versatile per la valutazione della robustezza dei modelli di machine learning di fronte a diversi tipi di attacchi avversariali. Attraverso l'implementazione di test avanzati e la creazione di un'infrastruttura modulare, abbiamo fornito un'utile risorsa per analizzare e migliorare la sicurezza dei modelli ML.
I principali risultati raggiunti includono:

- Un'accurata integrazione degli strumenti di ART per la simulazione di attacchi avversariali e la valutazione delle contromisure.
- La capacità di generare report dettagliati e informativi sull'efficacia dei modelli testati, evidenziando le loro vulnerabilità e punti di forza.
- Un'interfaccia utente intuitiva per facilitare l'interazione con la suite di test e la visualizzazione dei risultati.

Inoltre, il progetto ha dimostrato l'importanza di disporre di strumenti di testing avanzati per garantire la robustezza dei modelli ML, specialmente in applicazioni critiche dove la sicurezza è una priorità.

5.1 Sviluppi Futuri

Nonostante i risultati ottenuti, ci sono diverse aree di miglioramento e sviluppo futuro che potrebbero ampliare ulteriormente le capacità e le funzionalità della **ML Test Suite**. I principali sviluppi futuri includono:

- **Miglioramento dei Report:** Implementare un sistema di salvataggio locale dei grafici e delle statistiche dettagliate per facilitare l'analisi post-test e la condivisione dei risultati. Questo permetterà agli utenti di conservare e consultare in modo più efficiente i dati storici dei test.
- **Salvataggio dei Risultati in Blockchain:** Integrare una soluzione per il salvataggio dei risultati dei test sulla blockchain Ethereum. Questo garantirà l'immutabilità e la tracciabilità dei risultati, offrendo una ulteriore garanzia di integrità e trasparenza.
- **Supporto per Attacchi e Difese Aggiuntive:** Estendere la suite per includere supporto per un maggior numero di attacchi avversariali e difese, con misure di confronto dettagliate. Questo permetterà una valutazione più completa e accurata della robustezza dei modelli.
- **Parametri Aggiuntivi per Modelli Built-In:** Aggiungere supporto per parametri aggiuntivi per i modelli built-in della suite. Questo consentirà una personalizzazione più fine e una migliore adattabilità ai diversi scenari di test.
- **Utilizzo di Risorse Computazionali Cloud:** Implementare il supporto per risorse computazionali online attraverso piattaforme cloud come Google Cloud, AWS e Azure. Questo permetterà di scalare le operazioni di testing e di gestire grandi volumi di dati in modo più efficiente.

Questi sviluppi futuri non solo miglioreranno le capacità della **ML Test Suite**, ma contribuiranno anche a garantire una maggiore robustezza e affidabilità dei modelli di machine learning. L'adozione di queste migliorie sarà fondamentale per affrontare le sfide emergenti nel campo della sicurezza dei modelli ML e per supportare una comunità di ricerca sempre più vasta e diversificata.

Riferimenti

[1] T. AI, *Adversarial Robustness Toolkit (ART)*, Open-source library for enhancing machine learning model security against adversarial attacks., 2018. indirizzo: `https://github.com/Trusted-AI/adversarial-robustness-toolbox`.

[2] Facebook, Inc., *React Documentation*, Created by Jordan Walke., 2013. indirizzo: `https://reactjs.org/`.

[3] ECMAScript Standards Committee, *JavaScript Documentation*, Standard for JavaScript programming language., 1995. indirizzo: `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.

[4] Evan You, *Vite Documentation*, Creator of Vue.js., 2020. indirizzo: `https://vitejs.dev/`.

[5] Sebastián Ramírez, *FastAPI*, Born on 7th July 1985., 2018. indirizzo: `https://fastapi.tiangolo.com/`.

[6] Guido van Rossum, *Python*, The documentation for the Python programming language., 1991. indirizzo: `https://www.python.org/`.

[7] Consensys, *MetaMask*, Browser extension for interacting with Ethereum-based decentralized applications., 2016. indirizzo: `https://metamask.io`.